Modular parametric architecture design for ferry bilge systems

B.R. Herremans[1], T. Wilschut[2], M. Deul[3], B. Vink[3], R. Brouwer[3]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Ratio Computer Aided Systems Engineering B.V., Eindhoven, The Netherlands
[3]Damen Shipyards, Gorinchem, The Netherlands

Abstract: Damen shipyards has set out to develop a modular parametric ferry architecture to enable their nautical engineers to transition from the current implicit Configure-to-Order (CtO) design and realization process to an explicit CtO design and realization process. This paper presents a dependency structure matrix (DSM) based methodology to specify, visualize and analyze such an architecture. The methodology combines a clustered product DSM, sequenced parameter DSM and a clustered requirement DSM into a single multi domain matrix (MDM) that provides a clear structured overview of the dependencies between system components, design parameters, requirements and combinations thereof. The network of dependencies underlying this visualization are directly derived from an Elephant Specification Language (ESL) specification describing the system at hand. The methodology is illustrated using a ferry bilge system architecture design use case. The resulting MDM revealed a clear modular bilge system architecture and a nearly sequential design and configuration process.

Keywords: Product DSM, Parameter DSM, Requirement DSM, Modular architecture, Parametric design

1 Introduction

Damen Shipyards is a globally operating shipyard that, among others, designs and manufactures ferries. Figure 1 shows, for example, the Damen 6819E3 road ferry which can carry up to 300 people and 42 cars. The design of such ferries is usually tailored to specific routes and operational usage, even when they can reuse large parts of earlier designed solutions. Damen refers to their way of working as CTO with E, Configure to Order with a dedicated Engineering scope. In doing so, naval architects and designers and engineers try to maximize reuse of prior design solutions for functional elements of these ferries to reduce costs and risks. In doing so, they rely heavily on their personal experience, skills and tacit knowledge on ferry architectures and ferry legislation.

With the modern increase in complexity of ships, the introduction of novel (propulsion) technology and ever changing (local) legislation it has become increasingly difficult to rely on this implicit CtO approach only. Hence, Damen Shipyards has set out to realize an explicit CtO design and realization process using a properly and explicitly defined standardized modular parametric ferry architecture.

The development of this ferry architecture has been part of the New Advanced Value Added Innovative Ships (www.navais.eu) project, in which they adopted the Requirement-Function-Logical-Physical (RFLP) methodology (Kleiner, 2013, Li, 2020). The RFLP methodology dictates the breakdown and architecture design of a system following the requirements (R) that are linked to system functions (F) which are being filled by logical elements (L) that are realized as physical components (P). In applying this methodology, engineers, however, noticed that the resulting system breakdown and system architectures depend on the set of stated requirements and subjective decisions of engineers. Moreover, the functional and design dependencies between physical components of the system (P) are not explicitly accounted for in setting up the architecture. This is inconvenient as ferries sailing in different parts of the world are subject to different requirements dictated by local legislation and the resulting modules have quite a few dependencies. Damen prefers a modular parametric architecture that is independent of local legislation and contains modules that are mutually as independent as possible to achieve a high level of standardization and allow for shift configuration.

Dependency structure matrix (DSM) methods have been used in a wide variety of industries to model, analyze, design and improve system architectures. The excellent literature review of Browning (2015) provides over 500 references to DSM related works. A DSM is a square $N \times N$ matrix showing the dependencies among N system elements, such as, e.g., components, functions and (design) parameters (Eppinger, 2012). Multiple DSM can be assembled along the diagonal of a larger muti-domain-matrix (MDM), where the off-diagonal matrices are domain mapping matrices (DMMs) that show the dependencies between elements from the various domains (Eppinger, 2012).

This paper presents a dependency structure matrix (DSM) based method to develop a modular parametric architecture for ferries which enables naval architects and designers and engineers to quickly and explicitly configure a ferry system. The method combines a product DSM, a parameter DSM, and a requirement DSM in a product-parameter-requirement (PPR) multi-domain-matrix (MDM). The MDM is automatically derived from structured system (requirement) specifications following the method of Wilschut, 2018a which has evolved into the Elephant Specification Language (ESL) (Wilschut,

Figure 1. Road Ferry 6819E3 Amherst Islander II (courtesy of Damen Shipyards).

2018b,c). ESL is a language for creating highly structured systems specifications from which system architecture models can automatically be derived.

The product DSM, showing functional and design dependencies between system components, is clustered to determine a suitable modular system architecture. While the parameter DSM, showing design dependencies between design parameters, is sequenced to highlight the order in which one should determine the values of the different design parameters of the system components. The requirement DSM, showing design dependencies between requirements, is clustered to highlight which groups of requirements should be jointly resolved as they constrain the value of dependent design parameters.

The methodology is illustrated using a case study in which a modular parametric architecture for a ferry bilge system is designed and specified. A bilge system is a draining system responsible for (emergency) overboard discharge of undesired water from below main deck compartments. For example, during stormy weather conditions or accidental hull penetration. The bilge system is selected as a case example because it is a relatively simple safety system which is required in any ship of considerable size. Yet, as it is a safety critical system its configuration and detailed design is dependent on (local) legislation and overall ship geometry.

2 Ferry bilge systems

Figure 2 shows a schematic overview of a ferry bilge system. Black numbers indicate different hull compartment types which are separated by watertight walls. Black ones indicate the fore- and aft-peak compartments which are respectively the first and last compartments of the hull. Black twos indicate machinery space compartments which are spaces below deck where machinery is installed. Due to the essential operation of this machinery special requirements are in place regarding the bilge system configuration and design. Black threes indicate general space compartments which are non-machinery spaces such as a storage compartments. Bilge system requirements are less stringent for these compartments. A black four indicates the top deck which is used to park cars and trucks. Depending on the size and operational usage of the ship it may contain multiple machinery and general space compartments.

Blue numbers indicate the main components of the bilge system. A blue one indicates the main bilge line which runs along the length of the ship and collects water from separate compartments and for overboard discharge (blue five) by the bilge pumps (blue 4). The bilge pumps are also used to pressurize the firefighting system. That is, the bilge pumps pump water from the outside environment towards the fire hydrants.

Scattered across the vessel one may find multiple suction ends depending on the ship geometry. Two types of suction end are used. Blue twos indicate indirect suction ends which collect water and transport it to the main bilge line. They consist of a pipe with at the end a controllable butterfly valve, non-return valve, and a suction basket or bucket filter.

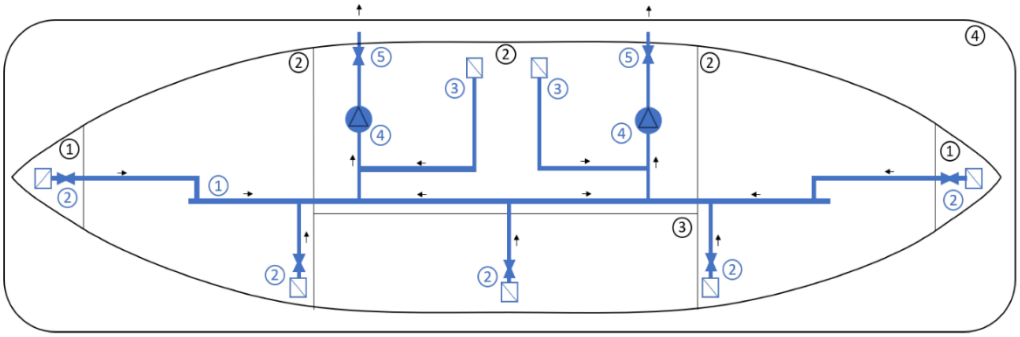

Figure 2. Simplified schematic top-view of a ferry bilge system. Black numbers indicate different hull compartment types, blue numbers indicate various pumping module types of the bilge system.

Alongside this suction end, a level switch is installed to measure the water level in a compartment. Blue threes indicate direct suction ends which are used in machinery space compartments. They bypass the main bilge line and are directly connect to the bilge pumps. They consist of the same type of components as the indirect suction ends. The butterfly valve can be remotely or manually controlled. The machinery compartment that houses the engine is equipped with an additional emergency bilge system which is redundant to the main bilge line system.

The configuration and detailed design of the components of the bilge line are to a large extent dictated by legislation. In particular, the International Convention for the Safety of Life at Sea (SOLAS) and the Classification Rules by Class Societies have a strong influence on the configuration and design.

SOLAS dictates minimum safety standards regarding the equipment, construction, and operation of ships that transport cargo or passengers. The Class Rules provides design rules and guidelines for safe ship design. A specific section is dedicated to the design of bilge systems for steel ships.

3 Method

In designing a bilge system, nautical engineers have to determine the system break down structure, the values of parameters that define the configuration, geometry and performance of the system, and ensure that requirements following from legislation are met. Additionally, they have to keep track of the dependencies between these elements as the design process usually requires multiple iterations. Therefore, the goal of the proposed method is to create a modular parametric architecture to reduce the number of needed iterations and simplify dependency tracking.

The method itself builds upon methods for the specification, visualization and analysis of the network of dependencies between system components, design parameters, and requirements. That is, it focusses on creating a model of the system architecture, defined as the mapping of a system's functions to the physical components within the system and to the dependencies between those components extend (Ulrich, 1995), extended with design parameters and requirements. In particular, the method is composed of the following steps:

1) **Specification** of the system decomposition, design parameters, and (functional) requirements using the Elephant Specification Language[1] (ESL) (Wilschut, 2018b,c) and subsequent automated derivation of dependencies between these elements.
2) **Visualization** of the network of dependencies between system components, design and configuration parameters, and requirements using a product-parameter-requirement (PPR) multi-domain-matrix (MDM).
3) **Analysis** of the network of dependencies between system components, design and configuration parameters, and requirements using a flow-based Markov clustering algorithm (Wilschut, 2017) and a flow-based sequencing algorithm based on tearing and Tarjan's strongly connected components algorithm (Tarjan, 1972).

In the following sections, each of these steps is explained in more detail, respectively.

3.1 Specification

One of the challenges in creating proper system architecture models is ensuring the consistency of the defined dependencies. Therefore, it is proposed to use ESL (Wilschut, 2018b,c), which is a formal language for creating highly structured multi-level system specifications from which system architecture models are automatically derived. Amongst others, ESL contains syntax for the definition of:

- **Components** that represent parts of the system decomposition tree.
- **Variables** that represent (functional) flows (Hirtz, 2002) that flow between components, such as electrical and mechanical energy, and properties of components, such as configuration, spatial and performance measures.
- **Goal requirements** that denote the functional purpose of one component with respect to another component. For example, the goal requirement *'g1: the battery shall provide power to the electric motor'* denotes that the purpose of *the battery* is to provide *power* to *the electric motor*.
- **Transformation requirements** that define the internal functional purpose of components. For example, the transformation requirement *'t1: the electric motor shall convert power into torque`* defines a functional (input-output) dependency between *power* and *torque* that must be realized by *the electric motor*.
- **Design requirements** that denote the requirements on, for example, the desired geometry, configuration and performance of components. For instance, the design requirement *'d1: pipe-diameter-p1 shall be equal to pipe-diameter-p2'* indicates the values of variables *'pipe-diameter-p1'* and *'pipe-diameter-p2'* need to be equal.

[1] https://docs.ratio-case.nl/manuals/

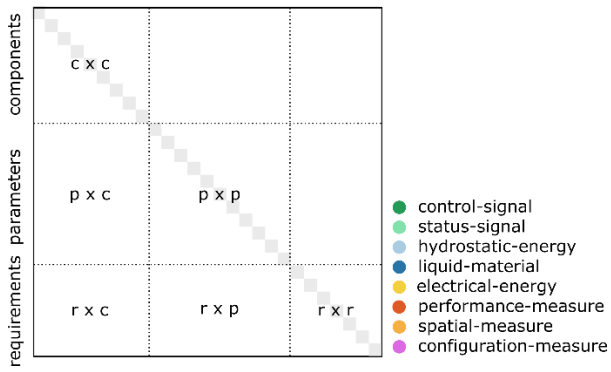

Figure 3. Schematic product – parameter – requirement multi-domain-matrix DSM showing functional and design dependencies. Functional dependencies are labelled with the labels control-signal, status-signal, hydrostatic-energy, liquid-material and electrical-energy. Design dependencies are labelled with the labels performance-measure, spatial-measure, and configuration-measure.

- **Relations** that denote (mathematical) design dependencies between variables. For example, the relation *'bilge-pump-capacity-calculation requires arguments main-bilge-diameter, ship-length, and number-of-bilge-pumps and returns argument bilge-pump-capacity*' denotes that the value of *bilge-pump-capacity* depends on the values of *arguments main-bilge-diameter, ship-length, and number-of-bilge-pumps*.

The ESL compiler automatically derives a dependency network based on the written system specifications following formal mathematical rules. For example, from goal requirement *g1* the ESL compiler derives a functional dependency between *the battery* and *the electric motor* which is quantified by the variable *power*. Additionally, mapping dependencies are derived between *the battery, g1* and *power,* as *the battery* is the subject of *g1* and should provide *power,* between *the*

electric motor, *g1* and *power,* as *the electric motor* is the indirect object of *g1* and receives power, and between *g1* and *power*, as *power* is the direct object of *g1* and quantifies the functional flow.

Similarly, the compiler derives functional and mapping dependencies from transformation-requirements and design and mapping dependencies from design-requirements and relations. In general, the ESL compiler derives multiple dependencies and mapping relations between a variety of elements throughout the system decomposition tree based on individual goal-requirements, transformation-requirements, design-requirements and relations following mathematical rules. This ensures consistency of the dependency network. The rules for deriving these dependencies are beyond the scope of this paper. The interested reader is referred to the ESL Reference Manual[2] in which these rules are fully documented.

3.2 Visualization

The ESL compiler yields a dependency network containing a variety of elements and dependencies. In this case, it is proposed to use a subset of those elements and dependencies to create a visualization tailored for nautical engineers of Damen to create a modular parametric architecture for the bilge system. Specifically, it is proposed to create an MDM schematically shown in Figure 3, which is composed of three DSMs and three DMMs:

1) A product DSM (c×c) that shows functional and design dependencies between all bilge system components that have to be designed or configured when realizing a ferry and all non-bilge system components of a ferry that have a functional or design dependency with one or more bilge system components. Both functional and design dependencies are included as the aim of the product DSM is to find modules of components that are as independent as possible. The product DSM is placed first as system components are the main focus of the designers.
   The bilge system is decomposed following a geometric decomposition up to of the shelve components, such as pumps and valves. The non-bilge system components are added as coarse-grained elements, such as fire-fighting (fifi) system, hull-compartments and power-supply, as they are not within the scope of the bilge system designers.
2) A parameter DSM (p×p) that shows design dependencies between the configuration, design and performance parameters of the bilge system which are subject to requirements following from legislation, such as the main bilge line diameter and minimum water flow speed, and those configuration, design and performance parameters on which the bilge system parameters depend, such as the length, width and gross-tonnage of the ferry. Parameters are placed second as the values of the system parameters is what the designers can actually influence.
3) A requirement DSM (r×r) that shows design dependencies between requirements derived from legislation that span the design and performance space for the bilge system. Requirements are placed third since these are formulated in terms of (design parameters). This is also the reason why functions are omitted from the MDM.

[2] https://docs.ratio-case.nl/reference/esl_reference

4) A parameter – component (p×c) domain mapping matrix (DMM) that shows which parameters define the configuration, design and performance of which components by means of mapping dependencies.
5) A requirement – parameter (r×p) DMM that shows which requirements bound the value of which parameters by means of mapping dependencies.
6) A requirement – component (r×c) DMM that shows which requirements influence the design of which components by means of mapping dependencies.

The dependencies displayed within the MDM are labelled to indicate the nature of the dependency. That is, functional dependencies are labelled as being control-signal, status-signal, hydrostatic-energy, liquid-material, or electrical-energy dependencies. These labels are a subset of the functional flows defined by functional basis (Hirtz, 2002) which flow through and between the bilge system components. Design dependencies are labelled as being a performance-measure, spatial-measure, or configuration-measure dependency to indicate that the design dependency relates to system performance, spatial design, or system configuration.

The resulting MDM provides a compact explicit overview of the bilge system architecture, extended with the parameters that characterize the configuration, spatial design, and performance of the system, the requirements derived from legislation that span the design and performance space, and the dependencies between them. This overview provides a structured basis for engineers to create a modular parametric architecture and simplify dependency tracking.

3.3 Analysis

Manually identifying a suitable modular parametric architecture and an optimal order in which to determine the values of the configuration, design and performance parameters from the proposed MDM may prove the be difficult due to the size of the matrix and the number of dependencies to be considered. Therefore, it is proposed to analyze the product and requirements DSM with a clustering algorithm and the parameter DSM with a sequencing algorithm.

The product DSM is clustered to highlight the modular architecture of the system. In this work, a flow-based hierarchical Markov clustering algorithm (Wilschut, 2017), implemented within open-source the Python RaGraph package[3], is used. This algorithm is specifically developed to cluster product DSMs in which so-called bus or integrative components are present that have many dependencies throughout the system.

Similarly, the requirement DSM is clustered using the same algorithm to identify groups of requirements that span the solution space of a single parameter. Hence, these requirements have to be resolved, verified and validated simultaneously.

The parameter DSM is sequenced to identify an optimal order in which to determine the values of the configuration, design and performance parameters. That is, a sequence that imposes a minimum number of feedback (upper-diagonal) dependencies such that the required number of design iterations is minimized. To find this sequence a flow-based sequencing algorithm based on tearing and Tarjan's strongly connected components algorithm (Tarjan, 1972) implemented within the Python RaGraph package is used.

4 Results

Figure 4 shows the system decomposition that reflects the system decomposition as defined by experts within the NAVAIS project. The decomposition has four levels. The first level contains the components *ship* and *outsideworld*. The ship is decomposed into the components *hull-compartments*, *bilge-system*, *fifi-system, emergency-bilge* and *power-supply*. A ferry is composed of many additional components. Those, however, have no direct functional or design dependencies with the bilge-system and are therefore omitted.

The bilge-system itself is decomposed into fifteen components following the general bilge system design as shown in Figure 2. In reality a bilge system may be composed of multiple *bilge pump modules* and *bilge fifi pump modules*, and *(in)direct suction ends* depending on the size of the ferry. Here only one instance of those modules are included as the dependencies of additional module instances will be exactly the same.

The *indirect* and *direct suction ends*, the *bilge pump* and *bilge fifi pump modules*, *bilge outlet*, and *bilge fifi outlet* are decomposed one level further to arrive at components that can be bought of-the-shelve.

This decomposition has been used as the basis for an ESL specification from which the PPR-MDM, show in Figure 5, has been derived. The hierarchical tree on the bottom left of Figure 5 represents the hierarchical structure as shown in Figure

[3] https://ragraph.ratio-case.nl/index.html

Figure 4. Specified system decomposition as defined by experts within the NAVAIS project .

4. The rows and columns of the product DSM (rows, columns 1-27) are labelled with the leaves components of the system decomposition tree. The black squares denote the component modules as defined within the NAVAIS project.

The parameter DSM (rows, columns 28, 49) contains the main performance, configuration, and spatial parameters of the ferry and bilge system that need to be determined before a detailed bilge system design can be made. Note that all of these parameters are properties of components within the system decomposition tree shown in Figure 4. The parameters that quantify the functional flows between components have been omitted as those are not directly subject to requirements derived from legislation. That is, requirements are, for example, formulated in terms of minimum pipe diameter and minimum bilge flow speed which will result in a minimum functional bilge flow.

The requirement DSM (rows, cols 50-56) contain the requirements derived from SOLAS and RCSS that directly affect the design of bilge system components, which is sufficient for setting up a modular parametric architecture for the bilge system.

Looking at the dependencies within the product DSM (rows, columns 1-27), one can see that the components have much more functional dependencies, labelled as control-signal, status-signal, hydrostatic-energy, liquid-material, or electrical-energy dependencies, than design dependencies, labelled as performance-measure, spatial-measure, or configuration-measure dependency. Moreover, one can observe many functional dependencies between the defined modules. As such, it seems that the functional dependencies derived from goal and transformation requirements should be leading over design-dependencies derived from design-requirements and relations in defining a modular parametric architecture[4]. Especially, as there seem to be quite a few bus components that have many dependencies throughout the system. This could explain the difficulties experienced within the NAVAIS project in which the design-requirements derived from legislation were leading in defining the system architecture.

[4] In practice, the presence of a functional dependency between two components implies the presence of design dependency between the two components, as one has to coordinate the design of these components to realize the functional interaction.

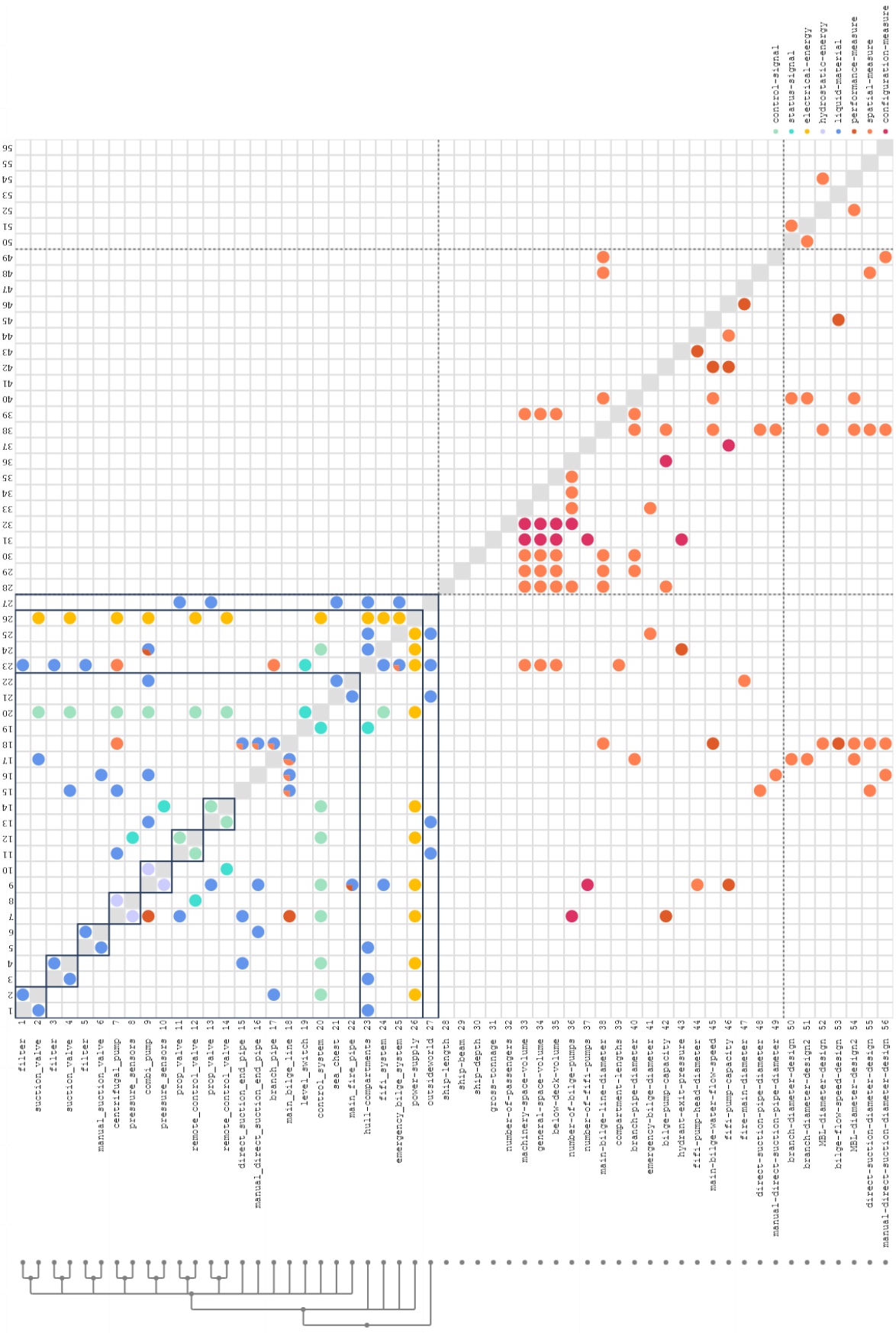

Figure 5. Product – parameter – requirement multi-domain-matrix (MDM), Showing the modular bilge system architecture defined by experts within the NAVAIS project.

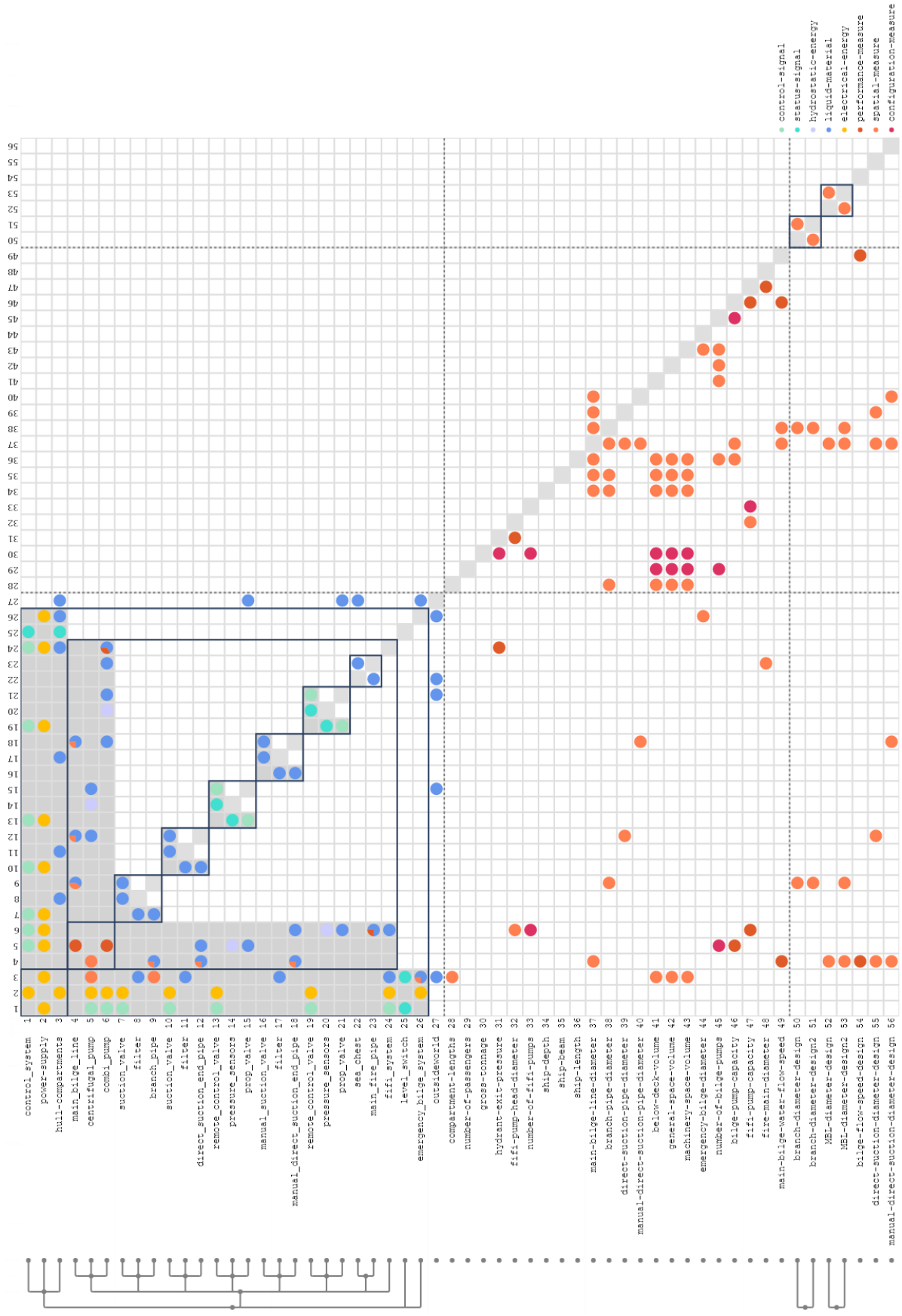

Figure 6. Clustered and sequenced product – parameter – requirement multi-domain-matrix (MDM).

The parameter DSM (rows, columns 28, 49) shows that most design dependencies between parameters are spatial dependencies between parameters that characterize the geometry of the ferry and bilge system components. However, a few configuration parameters, such as the *gross-tonnage* (row 31) and the *number of passengers* (row 32), (indirectly) influence quite a few spatial parameters. Moreover, one can observe that even without sequencing most dependencies are below the diagonal. Only six dependencies are placed above the diagonal. So, it seems feasible to achieve a configuration and design process with only a hand-full of feedback loops.

The requirement DSM (rows, cols 50-56) shows only a few dependencies between requirements. Hence only a few requirements define the solution space of the same parameter. The requirement – parameter DMM (rows 50-56, columns 28-49) and the requirement – component DMM shows that only five parameters and only four components are directly influenced by these requirements.

The parameter – component DMM (rows 28-49, columns 1-27) is fairly sparsely populated. That is, it contains quite a few empty rows and empty columns. An empty row indicates that the respective parameters is not a property of a leaf component of the decomposition tree which are shown in the product DSM, but of non-leaf components within the decomposition tree.

For example the *ship-depth* (row 30) and the *ship-length* (row 28). An empty column indicates that no properties of that component have to be determined before the detailed design process of the bilge system can start. Their properties can be determined in a later stage. For example, suitable *pressure sensors* (column 14) can be selected during the course of the detailed design and engineering process.

Figure 6 shows the restructured PPR-MDM in which the product DSM is clustered, the parameter DSM is sequenced, and the requirement DSM is clustered. The clustered product shows a hierarchical system decomposition. Bus clusters are indicated by a gray shaded background. The *outsideworld* component (row 28) has been excluded from the clustering routine as it is not part of the ferry. Hence it is not part of the ferry decomposition tree spanning rows 1-27.

On the first level, the ferry is decomposed into a bus-cluster (rows, columns 1-3) containing overarching bus components such as the *control-system*, *hull-compartments,* and *power-supply*, a large cluster (rows, columns 4-24) containing all components that are part of the bilge system and the firefighting system (*fifi system, row 24)*, and the leaf components *level-switch* (row 25) and *emergency bilge system* (row 26).

Internally, the bilge system contains a local bus-cluster (rows, columns 4-6) composed of the *main bilge line*, *centrifugal pump*, and *combi pump*, five clusters containing three components of which the first, second and fourth represent different types of suction ends and the third and fourth are bilge outlets, a cluster containing two components (rows, columns 23-24) which are used to take-in water from the *outsideworld* in case of fire, and the leaf component *fifi-system*.

Interestingly, the three bilge system bus components (rows, columns 4-6) have no direct functional dependencies only design dependencies. The first suction-end cluster (rows, columns 7-9) is an indirect suction end as it has no direct functional dependency with the *centrifugal pump* (row 5) or the *combi pump* (row 6), water is transferred via the *main* bilge line. The second (rows, columns 10-12) and the third section end cluster (rows, columns 16-19) are direct suction ends which are used in combination with the *centrifugal pump* (row 5) and the *combi pump* (row 6), respectively. Similarly, the first bilge outlet cluster (rows, columns 13-15) is used in combination with *centrifugal pump* (row 5), while the second bilge outlet cluster is used in combination with the *combi pump* (row 6). Depending on the *number-of-fifi-pumps* (row 33) and *number-of-bilge-pumps* (row 45), a real ferry bilge system may contain multiple types of suction ends and bilge-outlets.

To some extent, the same system structure can be found within the product DSM of Figure 5. That is, the first, second, third, sixth and seventh cluster of two components, shown in Figure 5, correspond with the five clusters containing three components. As a result of explicitly accounting for bus components, connecting pipe sections, such as a *branch pipe* (row 9), have been added to the respective clusters, increasing their size from two to three and reducing the number of intra-cluster dependencies significantly.

In Figure 6, the sequenced parameter DSM (rows, columns 28-29) only contains three upper diagonal dependencies. These feedback marks are the result of requirements that state that the diameters of the various pipe sections should be equal. As a consequence, the *main bilge line diameter* (row 37), *branch pipe diameter* (row 38)*, direct suction end diameter* (row 39), and *manual direct suction pipe diameter* (row 40) have bi-directional design dependencies. In practice, however, the *main bilge line diameter* is usually leading in designing the pipe sections.

As such, the sequenced process DSM reveals a nearly sequential bilge system configuration and design process. Note that the rows of parameters *compartment-lengths* (row 28), *number-of-passengers* (row 29), *gross-tonnage* (row 30), *ship-*

depth (34), *ship-beam* (row 35), and *ship-depth* (row 36) are empty within the parameter DSM. This implies that these six parameters are input to the bilge-system design process. All other parameters (indirectly) follow from them.

The clustered requirement DSM reveals two clusters of requirements. The first cluster (rows, columns 50-51) spans the design space for the *branch pipe diameter* (row 38), while the second cluster (rows, columns 52-53) spans the design space for the *main bilge line diameter* (row 37). Note that requirement *bilge-flow-speed-design* (row 54) relates to the parameter *main-bilge-line-water-flow-speeds* which is the final (output) parameter within the sequenced bilge system configuration and design process.

5 Conclusion

This paper presents a dependency structure matrix (DSM) based methodology to specify, visualize, and analyze a modular parametric ferry architecture. The methodology combines a clustered product DSM, sequenced parameter DSM and a clustered requirement DSM into a single multi domain matrix (MDM) that provides a clear structured overview of the dependencies between system components, design parameters, requirements and combinations thereof. This overview allows for an early-stage review of the design for consistency and robustness, so it helps preventing costly redesign effort later in the process. The network of dependencies underlying this visualization are directly derived from an ESL (Wilschut, 2018b,c) specification describing the system at hand.

The bilge system architecture design case study showed that ESL can effectively be used to describe a system and that the derived network of dependencies can be effectively used for architecture visualization and analysis. In particular, the results showed that the modular architecture defined within the NAVAIS project can be improved by explicitly accounting for the presence of bus elements and including the connection pipe sections to the clusters that represent the various types of suction ends and bilge outlet. Additionally, the sequenced parameter DSM revealed the possibility to setup a nearly sequential configuration and design process for the bilge system to minimize rework.

References

Browning, T. R., 2015. Design structure matrix extensions and innovations: a survey and new opportunities. *IEEE Transactions on engineering management*, 63(1), 27-52.

Doerry, N., 2009. Using the design structure matrix to plan complex design projects. In *Proceedings of the ASNE Intelligent Ships Symposium*, Philadelphia, Pennsylvania, USA.

Eppinger, S. D., Browning, T. R., 2012. *Design structure matrix methods and applications.* MIT press, Massachusetts, USA.

Hirao, A., Koga, T., and Aoyama, K., 2010. Planning Support of Initial Design Process Based on Grouping and Ordering of Tasks – Design Example of an Integrated Circuit . In*: DSM 2010: Proceedings of the 12th International DSM Conference*, Cambridge, UK, 83–96.

Hirtz J., Stone R.B., McAdams D.A., Szykman S., Wood K.L., 2002. A functional basis for engineering design: reconciling and evolving previous efforts. *Research in Engineering Design,* 13(2), 65–82.

Kleiner, S., Kramer, C., 2013. Model based design with systems engineering based on RFLP using V6. *Smart Product Engineering*. Springer, Berlin, Heidelberg. 93-102.

Li, T., Lockett, H. and Lawson, C., 2020. Using requirement-functional-logical-physical models to support early assembly process planning for complex aircraft systems integration. *Journal of Manufacturing Systems,* 54, 242-257.

Maier, J. F., Eckert, C. M. & Clarkson, J. P. Model, 2017. Model granularity in engineering design – concepts and framework. *Design Science*, 3, 1–29.

Tarjan, R., 1972. Depth-first search and linear graph algorithms. *SIAM journal on computing 1(2),* 146-160.

Tilstra A.H., Seepersad C.C., Wood K.L., 2012. A high-definition design structure matrix (HDDSM) for the quantitative assessment of product architecture. *Journal of Engineering Design,* 23(10–11), 767–789.

Ulrich, K., 1995. The role of product architecture in the manufacturing firm. *Research Policy*, 24(3), 419-440.

Wilschut, T., Etman, L.F.P., Rooda, J.E., Adan, I. J. B. F., 2017. Multilevel flow-based Markov clustering for design structure matrices. *Journal of Mechanical Design,* 139(12), 121402.

Wilschut, T., Etman, L.F.P., Rooda, J.E., Vogel, J.A., 2018a. Generation of a function-component-parameter multi-domain matrix from structured textual function specifications. *Research in Engineering Design,* 29, 531–546.

Wilschut, T., 2018b. *System specification and design structuring methods for a lock product platform.* PhD dissertation. Eindhoven University of Technology.

Wilschut, T., Etman, L. F. P., Rooda, J. E., & Vogel, J. A., 2018c. Multi-level function specification and architecture analysis using ESL: A lock renovation pilot study. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* Vol. 51845, V007T06A038. American Society of Mechanical Engineers.

Contact: T. Wilschut, Ratio Computer Aided Systems Engineering B.V., Eindhoven, The Netherlands, t.wilschut@ratio-case.nl